CS112 - Fall 2022

Lab01

Instructor: Paul Haskell

INTRODUCTION:

Git and GitHub are different but closely related software tools that are used to store and manage software source code files. You can imagine that if you are working on a big software project, especially a project with other people, it might be handy to be able to retrieve a version of what the software looked like a week ago or a month ago. Maybe some recent changes broke the software, maybe you need to know what changed in the past month, maybe you need to know who made which changes, etc.

- **Git** is a file management software tool that you run on your own computer. It lets you create a "project repository", add files to the repository, update the repository with changes to files, retrieve older versions, retrieve a working copy of the repository on your own local computer, etc.
- A **repository** is basically smart storage that stores the current version of a set of files but also all past versions. That way, if some change accidentally breaks the software, the previous version can be retrieved.
- **GitHub** is a Cloud based repository manager. It stores repositories in the cloud, lets the owner determine who can access repositories, etc. It has a bunch of other development tools that we will not use in CS112. GitHub provides extremely reliable storage, so if your computer breaks or is stolen, you will not lose any of your software that was pushed to the GitHub cloud archive.

For the CS112 class, each of you will work with two repositories:

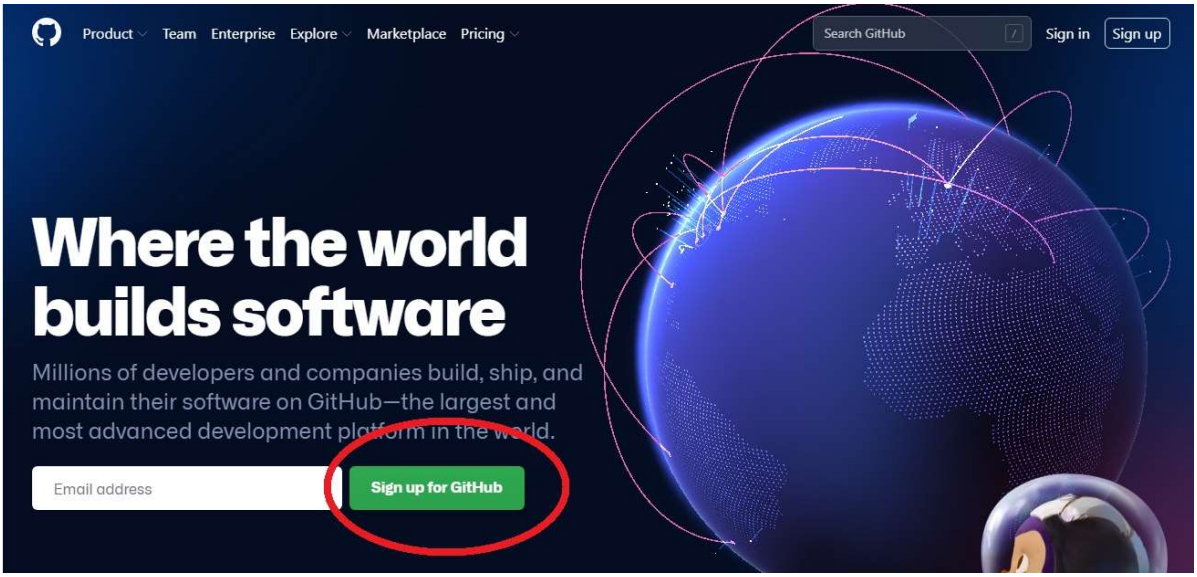
- A shared one called "CourseInfo" that the instructor and TA's will populate with documents, sample software, assignments, etc. You each will make a local copy on your own computer, and you will update your local copy when I tell you there are updates on the CourseInfo repository's cloud server. All students can read this repository, and the TA's and instructor can read and write it.
- An individual repository for each of you that you will use to turn in your work. You will write, debug, and test software on your local machine and then will "push" it to your individual repository on the Cloud server. The TA's and instructor then will fetch your software, test it, review it, and grade it. The TA's and instructor will fetch each of your repositories at the deadline for each assignment. Anything pushed after the deadline will not be retrieved and will be deemed to be not completed. Other students cannot read or write your repository.

LAB01:

For this lab, you will first set up Git and GitHub on your computer, download the CourseInfo repository, set up your personal repository, and upload a few files.

GITHUB SETUP:

- If you do not yet have a GitHub account tied to your USF email, set one up:
- Browse to https://www.github.com and click the "Sign Up for GitHub" button



Fill out the requested information and complete the registration. You will probably continue to use your GitHub UserID throughout your time at college, sharing it with professors and fellow students, so please pick a UserID you will be happy to share publicly. Of course, do not share your GitHub password. You do not need any of the other services GitHub offers, and you can use a Free account.

After you create your account, GitHub will give you choices on what to do next. Don't do anything yet!

- Once you have completed your registration, email your GitHub UserID to phaskell@usfca.edu (or if you are in class, simply walk up and tell Paul). Paul will add you to the GitHub "Organization" for this class and will give you permission to read the "CourseInfo" repository. You will get an email from GitHub asking you to join the "CS112-phaskell" Organization. Click the link to accept. On the github.com website, refresh your browser. If you are asked to join the "CS112-phaskell" Organization and the "CourseInfo" repository, accept those invites.

GIT SOFTWARE SETUP:

While waiting for Paul to add you to the GitHub Organization, move on to this step.

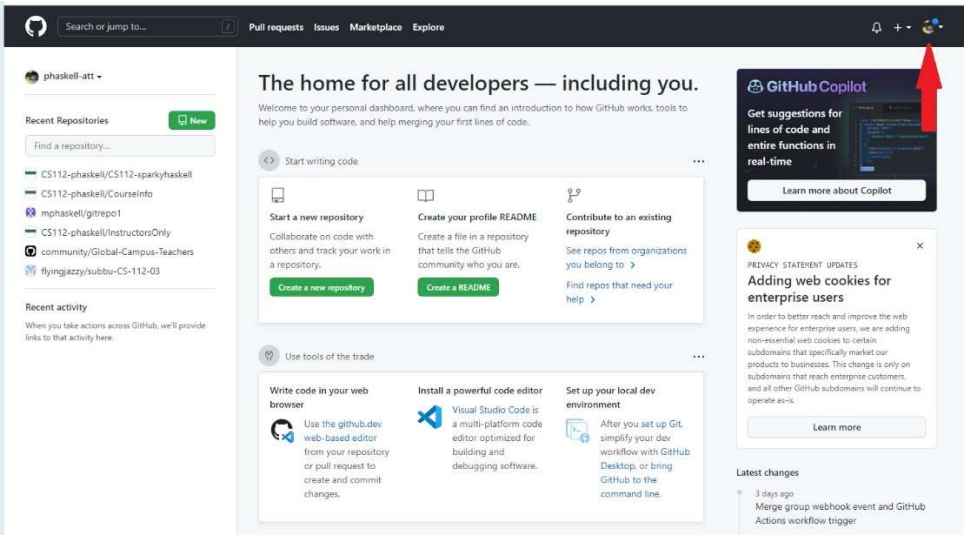
- Install the “git” command line tool on your personal computer.
    - https://docs.github.com/en/get-started/quickstart/set-up-git
    - https://git-scm.com/downloads
    - During installation, you may be asked to select a “Credential Manager”. This saves your GitHub password, so you do not need to retype it every time you run a git command. To set up the Credential Manager, during git installation:
        - On Windows, select “Git Credential Manager core”
        - On Mac, select “OSX keychain” if asked
        - On Linux, there are more steps to do: contact the instructor or TA’s
- Configure git with your personal information. From a terminal window, type:
    - `git config --global user.email <<your email address>>`
    - `git config --global user.name "<<your firsname & lastname inside doublequotes>>"`
- Git and GitHub support many features that we will not need in CS112, for example:
    - They support “branches”, in which multiple separate working versions of a software project can be created, modified independently, tested, and reviewed before they are merged back to the main software.
    - GitHub supports multi-person reviews (“pull requests”) of software changes, so changes can be viewed and discussed by experts before they are committed to the official repository.

AUTHENTICATE YOUR LOCAL git PROGRAM WITH GitHub

The first time you run a git command that pulls or pushes data to GitHub, GitHub will authenticate your computer, hopefully permanently. In the future, you can simply run git commands from your computer. But the first time, there are some extra steps.

- GitHub might simply ask you to authenticate from your web browser. If so, GitHub will open a new tab on your browser. Simply enter your GitHub password when prompted, follow the instructions, and you should be set.
- When you run your first git command, you might be prompted to enter your GitHub password. But if you enter your password, GitHub will tell you that passwords are no longer accepted. Instead, you must enter a "personal access token". Here is how to get this token.
    - In your web browser, log into GitHub.

- Click on the little round button in the upper right, and select the "Settings" option



  - On the bottom of the left side, select "Developer Settings"
  - On the left, select "Personal access tokens"
  - Set the "Expiration" to be some time after the class ends: December 31st?
  - Select at least the following "scopes": repo, admin:org, user, delete_repo
  - Then select "Generate Token". Copy the resulting token and paste it into a text document, so you have a copy.
- Now when git prompts you for your password, paste in the Personal Access Token. This should let the command execute successfully. And the token should be stored in your "Credential Manager", so from now on, you can run git commands without being asked for a password.

DOWNLOAD THE CourseInfo REPOSITORY TO YOUR COMPUTER:

Do this after you have accepted the invite to the CourseInfo repository.

- Run a terminal window ("command window" on Windows).
- Create a directory on your personal PC to store the shared course info and also your coursework:
  - In a convenient place on your computer's filesystem, make a directory called "**CS112**": type: `mkdir CS112`
  - Now have the terminal window enter the directory: type: `cd CS112`
  - Clone material in the CourseInfo repository to your local computer: from the **CS112** directory, in your terminal or command window, type:

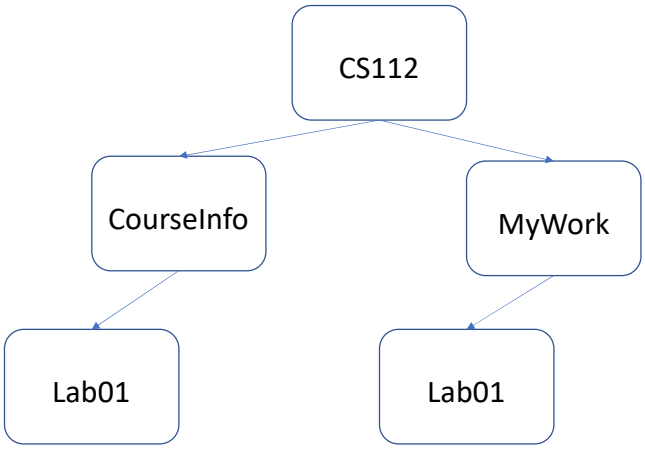
```
git clone https://github.com/CS112-phaskell/CourseInfo CourseInfo
```

    - "clone" tells git to copy all files in the cloud repository to your local machine
    - https://github.com/CS112-phaskell/CourseInfo is the name of the shared repository for this class

        - "**CourseInfo**" is the subdirectory inside the **CS112** directory on your personal computer that will store the shared course information
        - If you are asked to authenticate with your browser or with your Personal Access Token, do so.
    - Now from your terminal window, file explorer, or any other application, you can browse the **CourseInfo** directory on your local computer to see what is there.
    - Look inside the **CourseInfo** directory for the "**Lab01**" subdirectory. In the **Lab01** directory find the file called "P**aulsFavoritePizza.pdf**" and read the file to see what Paul's favorite pizza is. You will need that information in a few minutes.

SET UP A DIRECTORY ON YOUR PERSONAL COMPUTER TO STORE YOUR COURSEWORK

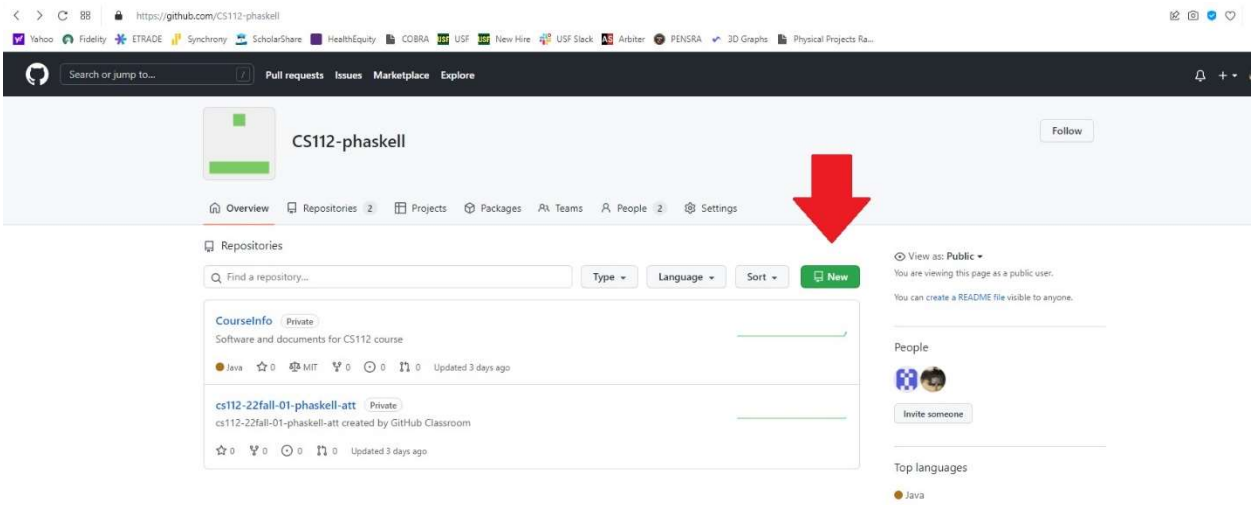
- cd to your **CS112** directory. Make a directory called "**MyWork**": `mkdir MyWork`
- `cd MyWork`
- Create a subdirectory called **Lab01**. (In later meetings you will create directories called Lab02, Lab03, …)



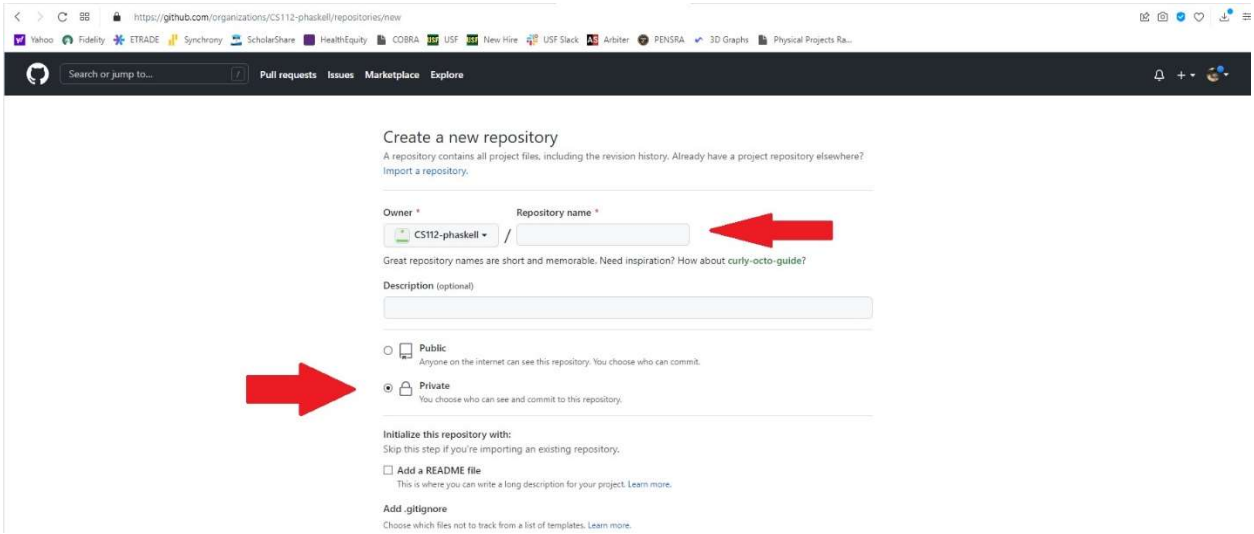
- Inside your **Lab01** directory, create a file called "**MyInfo.pdf"** (use whatever document editing you like: Microsoft Word, etc). In MyInfo,pdf, please enter:
    - Your name
    - Your USF login ID
    - Your GitHub ID
    - Which section of the class you are taking (9:30am or 1pm)
    - A few lines of text. First, "Professor Haskell's favorite pizza is <<the correct answer>>"
    - Then, enter some interesting but not overly personal information about yourself, to help the TA's and Paul learn all your names more quickly. Can you ride a unicycle? Do you like black licorice? Have a pet cat named Steph Curry?
    - Please include a photo of your face (to help the TA's and Paul learn all your names)

SET UP YOUR PERSONAL GITHUB REPOSITORY AND TIE IT TO YOUR “MyWork” DIRECTORY

- (If you closed your github.com webpage already, log into the github.com website.)
- (If your github.com screen is not saying “Create your first CS112-phaskell repository, then click on the little round button in the upper right, select the “Your Organizations” option, and select “CS112-phaskell”.)
- Click the “New” or “Create a new repository” button to create a new repository.



- Name your repository “CS112-<<GitHub UserID>>” (e.g. CS112-stephcurry)
- Make it PRIVATE – not public
- Make sure that the Owner is “CS112-phaskell”



- Click the “Create Repository” button. Don’t close the browser window! And don’t run any of the offered “quick setup” steps.
- In a terminal window, change to the **MyWork** directory (If you are in the **Lab01** directory, type `cd ..`)
    - Type: `git init -b main`

- Type: `git add .`
- Now "commit" your added directories and files: type (include the double-quote marks)
  - `git commit –m "My first commit for Lab01"`
  - The words after the '-m' are a comment that will be associated with this change to your repository
- On the webpage with your GitHub repository, grab the URL for your repository from the webpage titlebar: probably something like https://github.com/CS112-phaskell/CS112-YourUserID
- Now type: `git remote add origin <<TheURLYouCopied>>`
- Now send your file to your repository storage in the cloud: type: `git push origin main`
- That's it! You now have a personal repository set up for the class, and you have uploaded your **Lab01** directory.

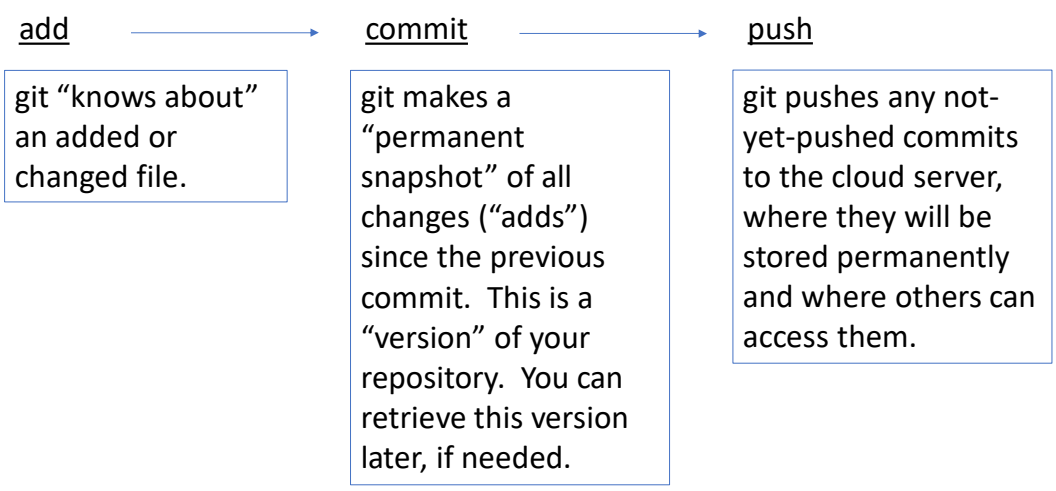
NOW THAT YOUR REPOSITORY IS SET UP…

Now that your repository is set up, it is easier to add new files to your repository, update your repository with changes to make to existing files, etc.

- In your **Lab01** directory, create a new file called **MyComputerInfo.pdf** . This file should contain the information about your computer that we collected during class. Now you will add this file to your existing repository.
- In a terminal window, in the **Lab01** directory, type: `git add MyComputerInfo.pdf`
  - You can add several files at once and can add a directory (and all its subdirectories and files) instead of adding just one file.
- Now "commit" that file, by typing:

```
git commit –m "Adding second file to repository"
```

  - If you don't include the "-m Message" option, git will open a text editor so that you enter a comment.
- Now "push" this commit to the cloud server by typing: `git push origin main`
  - You can make "origin" the default destination and "main" the default branch by running the command: `git push --set-upstream origin main`
  - Going forward, you can now just type: `git push`



CONCLUSION:

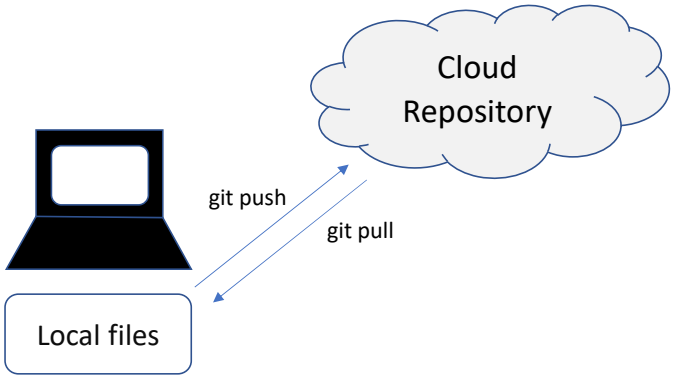
The deadline for completion of Lab01 is 11:59pm Saturday August 27.

The main goals of this lab are for you to install software that will be required for the class, to set up your GitHub repositories, and to follow the lab instructions accurately to demonstrate you succeeded in the first two goals. Future labs will have more involved scoring rubrics, and a portion of their scoring will be based on instructor-determined standards of Design Quality and Code Quality. For this first lab, the scoring rubric is simpler:

Task	Score, points
Created a personal GitHub repository with the correct name	1
Created properly named subdirectory	1
Created properly named **MyInfo.pdf**	1
Created properly named **MyComputerInfo.pdf**	1
MyInfo.pdf and **MyComputerInfo.pdf** contain all the requested information	5

NOTES FOR CS112

- When there are updates to the CourseInfo repository, I will let you know via Piazza and/or in class.
- You must re-fetch the repository to see the updated and new files.
    - To do this, from a terminal window, change directories to the CourseInfo directory, and type: `git pull`



- From the github.com website, you can view your repositories, add and delete files, etc. You can also do this using the git command on your personal computer. If you make changes in both places, the two views of your repository can get out of sync, and it will take some work to clean up. I suggest you use the github.com website only for viewing, and make all changes via git commands on your computer.

NOTES ON GIT

- You can add several files before committing a change, and can do several commits before pushing to the GitHub server.
- To see a list of files that are changed but not yet committed, or committed but not yet pushed, type: `git status`
- To see a list of all of a repository's "git commit" comments and their times, type: `git log`
- You cannot add, commit, or push empty directories. To add a directory to your repository, simply create an empty file in the directory (e.g. named **empty**).
- To get help on git, run: `git help` or: `git help <<commandName>>`
- You can remove a file from your repository by typing: `git rm <<NameOfTheFile>>`, and then going through the "commit" and "push" cycle.
- There is a long list of git commands at: https://git-scm.com/docs
- Please practice using git to add files and directories to your repository, to change files and update them, etc. Create/add/commit/push a new directory for practicing rather than the **Lab01** directory. We will look at and grade only the contents of the **Lab01** directory.

READ ON YOUR OWN - how to get training on GitHub

- Accept this course's "GitHub training assignment" at https://classroom.github.com/a/k34AfzSj. The README file in the resulting repository has useful information and useful links. Note that we will not be using many of the features of GitHub in CS112.
- If you would like more information on GitHub, check out these resources. You do not have to read all of these! Feel free to browse and see if you find anything useful, or just save these links if you actually run into problems:

    - https://docs.github.com/en/get-started/signing-up-for-github/signing-up-for-a-new-github-account
    - https://docs.github.com/en/get-started/quickstart/git-and-github-learning-resources
        - Look on the left side of the screen for documents on a wide variety of GitHub related topics
    - https://docs.github.com/en
    - https://skills.github.com/
    - https://githubtraining.github.io/training-manual/#/01_getting_ready_for_class
- For this course, you can **ignore** the following GitHub features and capabilities:
    - Issues (bug reports)
    - Actions (software "lifecycle management")
    - Integrations
    - Discussions
    - GitHub Pages
    - Pull requests (requests for someone else to review your code)
    - Branches (separate "tracks" or "branches" of software that can be developed, reviewed, and tested independently before being merged back to the "main branch")